CLAIMS

What is claimed is:

1. A programmable interpreter comprising:

means for receiving a command input stream, said command input stream having a command identifier;

means for encoding said command identifier into a corresponding processing component identifier; and

means for executing a processing component identified by said processing component identifier.

2. The programmable interpreter as claimed in Claim 1 further including means for pushing an argument onto a stack, said argument used as an input to said processing component identified by said processing component identifier.

3. The programmable interpreter as claimed in Claim 1 wherein said means for encoding further includes means for generating an execution stream for storage of said processing component identifier and associated arguments.

4. The programmable interpreter as claimed in Claim 1 further including means for popping an argument from a stack, said argument used as an input to said processing component identified by said processing component identifier.

5. The programmable interpreter as claimed in Claim 1 further including means for pushing a result of the execution of said processing component onto a stack.

6. The programmable interpreter as claimed in Claim 3 wherein said means for encoding further includes means for pointing to the first item

associated with said processing component stored in said execution stream.

7. The programmable interpreter as claimed in Claim 3 wherein said means for encoding further includes means for pointing to the first item associated with a second processing component stored in said execution stream.

8. The programmable interpreter as claimed in Claim 1 wherein said means for executing further includes means for recursively executing a processing component.

9. The programmable interpreter as claimed in Claim 3 further including means for interpreting said execution stream.

10. The programmable interpreter as claimed in Claim 3 wherein said execution stream is stored in random access memory.

11. In a programmable interpreter, a process for interpreting a command stream comprising the steps of:

receiving a command input stream, said command input stream having a command identifier;

encoding said command identifier into a corresponding processing component identifier; and

executing a processing component identified by said processing component identifier.

12. The process as claimed in Claim 11 further including the step of pushing an argument onto a stack, said argument used as an input to said processing component identified by said processing component identifier.

13. The process as claimed in Claim 11 wherein said step of encoding further includes a step of generating an execution stream for storing said processing component identifier and associated arguments.

14. The process as claimed in Claim 11 further including a step of popping an argument from a stack, said argument used as an input to said processing component identified by said processing component identifier.

15. The process as claimed in Claim 11 further including a step of pushing a result of the execution of said processing component onto a stack.

16. The process as claimed in Claim 13 wherein said step of encoding further includes a step of pointing to the first item associated with said processing component stored in said execution stream.

17. The process as claimed in Claim 13 wherein said step of encoding further includes a step of pointing to the first item associated with a second processing component stored in said execution stream.

18. The process as claimed in Claim 11 wherein said step of executing further includes a step of recursively executing a processing component.

19. The process as claimed in Claim 13 further including a step of interpreting said execution stream.

20. The process as claimed in Claim 11 further including a step of parsing said command input stream.